5. リモート・プログラミング

5.1 GPIB コマンド・インデックス

この GPIB コマンド・インデックスは、5 章の GPIB コマンド索引として活用して下さい。

GPIBコマンド 参照ページ

APPLICATION コマンド
:CALCulate2:DOMInant[:STATe] 5-39
:CALCulate2:LEVel:TYPE 5-39
:CALCulate2:TPOWer[:DATA]? 5-39
:CALCulate3:BANDwidth:DATA? 5-39
:CALCulate3:BANDwidth:K 5-39
:CALCulate3:BANDwidth:KR 5-39
:CALCulate3:BANDwidth:TYPE 5-39
:CALCulate3:BANDwidth:XDB 5-39
:CALCulate3:BANDwidth:YDB 5-39
:CALCulate3:BANDwidth[:STATe] 5-39
:CALCulate3:DATA? 5-39
:CALCulate3:POINts? 5-39
:DISPlay[:WINDow]:DUAL[:STATe] 5-39
:DISPlay[:WINDow]:LIST:ALL 5-39
:DISPlay[:WINDow]:LIST:CURRent 5-39
:DISPlay[:WINDow]:LIST:TYPE 5-39
:DISPlay[:WINDow]:LIST[:STATe] 5-39
:DISPlay[:WINDow]:TRACe 5-39
:DISPlay[:WINDow]:TRACe:SUPErimpose 5-39
{:MEASure | FETCh } [:SCALar]:LENGth:COHerency:ALPHa? 5-39
{:MEASure | FETCh } [:SCALar]:LENGth:COHerency:BETA? 5-39
{:MEASure|:FETCh}[:SCALar]:LENGth:COHerency:PEAK:RANGe 5-39
{:MEASure|:FETCh}[:SCALar]:LENGth:COHerency:PEAK:RANGe:STARt 5-39
{:MEASure|:FETCh}[:SCALar]:LENGth:COHerency:PEAK:RANGe:STOP 5-39
{:MEASure|:FETCh}[:SCALar]:LENGth:COHerency:POINt 5-39
CURSOR コマンド
:DISPlay[:WINDow]:MARKer:ACTIve 5-40
:DISPlay[:WINDow]:MARKer:AOFF 5-40
:DISPlay[:WINDow]:MARKer:DATA? 5-40
:DISPlay[:WINDow]:MARKer:MAXimum:LEFT 5-40
:DISPlay[:WINDow]:MARKer:MAXimum:RIGHt 5-40
:DISPlay[:WINDow]:MARKer:MAXimum:SEcond 5-40
:DISPlay[:WINDow]:MARKer:MODE 5-40
:DISPlay[:WINDow]:MARKer1:STATe 5-40
:DISPlay[:WINDow]:MARKer1:WAVelength 5-40
:DISPlay[:WINDow]:MARKer2:STATe 5-40
:DISPlay[:WINDow]:MARKer2:WAVelength 5-40
:DISPlay[:WINDow]:MARKer3:POWer 5-40
:DISPlay[:WINDow]:MARKer3:STATe 5-40
:DISPlay[:WINDow]:MARKer4:POWer 5-40

:DISPlay[:WINDow]:MARKer4:STATe 5-40
GPIB 機能コマンド
:FORMat:ASCii[:Y]:DATA 5-42
:FORMat:BORDer 5-42
:FORMat:DATA 5-42
:STATus:DEVice:ENABle 5-42
:STATus:OPERation:ENABle 5-42
:STATus:OPERation[:EVENt]? 5-42
:STATus:PRESet 5-42
:STATus:QUEStionable:ENABle 5-42
:STATus:QUEStionable[:EVENt]? 5-42
:SYSTem:ERRor:ALL? 5-42
:SYSTem:ERRor? 5-42
:STATus:DEVice 5-42
MEASURE コマンド
:ABORt 5-36
:INITiate:CONTinuous 5-36
:INITiate:IMMediate 5-36
:SENSe:CORRection:DEVice 5-36
RANGE コマンド
:SENSe:MODE 5-36
:SENSe:RANGe 5-36
:SENSe:RANGe:AUTO 5-36
:SENSe:RANGe:PULSe 5-36
SAVE/LOAD コマンド
:FILE:LOAD 5-41
:FILE:STORe 5-41
SCALE コマンド
:DISPlay[:WINDow]:TRACe 5-38
:DISPlay[:WINDow]:TRACe:ALL[:SCALe]:AUTO 5-38
:DISPlay[:WINDow]:TRACe:GRAPhics:GRID[:STATe] 5-38
:DISPlay[:WINDow]:TRACe:Y[:SCALe]:PDIVision 5-38
:DISPlay[:WINDow]:TRACe:Y[:SCALe]:RLEVel 5-38
:UNIT:POWer 5-38
SETUP コマンド
:CALCulate1:CAVErage:COUNt 5-37
:CALCulate1:CAVErage:TYPE 5-37
:CALCulate1:CAVErage[:STATe] 5-37
:CALCulate2:SMOothing:APERture 5-37
:CALCulate2:SMOothing:THReshold 5-37
:CALCulate2:SMOothing[:STATe] 5-37
:DISPlay[:WINDow]:TRACe:FULL:POINts? 5-37
:DISPlay[:WINDow]:TRACe:POINts? 5-37
:DISPlay[:WINDow]:TRACe:X:DATA? 5-37
:DISPlay[:WINDow]:TRACe:X:FULL:DATA? 5-37
:DISPlay[:WINDow]:TRACe:X[:SCALe]:ACTIve 5-37
:DISPlay[:WINDow]:TRACe:X[:SCALe]:CENTer 5-37
:DISPlay[:WINDow]:TRACe:X[:SCALe]:CENTer:MARKer 5-37
:DISPlay[:WINDow]:TRACe:X[:SCALe]:CENTer:PEAK 5-37
:DISPlay[:WINDow]:TRACe:X[:SCALe]:LEFT 5-37
:DISPlay[:WINDow]:TRACe:X[:SCALe]:RIGH 5-37

:DISPlay[:WINDow]:TRACe:X[:SCALe]:SPAN 5-37
:DISPlay[:WINDow]:TRACe:X[:SCALe]:SPAN:MARKer 5-37
:DISPlay[:WINDow]:TRACe[:Y]:DATA? 5-37
:DISPlay[:WINDow]:TRACe[:Y]:FULL:DATA? 5-37
:SENSe:POWer:RLEVel 5-37
:SENSe:POWer:RLEVel:AUTO 5-37
:SENSe:SWEep:POINts 5-37
{:MEASure|FETCh}[:SCALar]:PEAK? 5-37
{:MEASure|FETCh}[:SCALar}:PEAK:SEcond? 5-37
SYSTEM コマンド
:DISPlay[:WINDow]:TEXT:DATA 5-41
:SYSTem:DATE 5-41
:SYSTem:HALT 5-41
:SYSTem:OPTion? 5-41
:SYSTem:PRESet 5-41
:SYSTem:TIME 5-41
共通コマンド
*CLS 5-25
*DDT 5-26
*DMC 5-27
*EMC 5-28
*ESE 5-28
*ESR? 5-29
*GMC? 5-30
*IDN? 5-30
*LMC? 5-30
*OPC 5-31
*PMC 5-31
*RCL 5-31
*RST 5-32
*SAV 5-32
*SRE 5-33
*STB? 5-34
*TRG 5-35
*TST? 5-35
*WAI 5-35

5.2 GPIB リモート・プログラミング

本器は、IEEE 規格 488.1-1987 および 488.2-1987 に準拠した GPIB(General Purpose Interface Bus) を標準装備し、外部コントローラによるリモート・コントロールが可能です。

以下、GPIB リモート・コントロール機能を用いたコントロール方法について説明します。

5.2.1 GPIB とは

GPIB(General Purpose Interface Bus) は、コンピュータと計測器を統合する高性能のバスを提供します。

この GPIB の動作は IEEE 規格 488.1-1987 によって定義されています。GPIB はバス構造のインタフェースのため、各機器が固有の互いに異なる機器アドレスを持つことによって、特定の機器を指定します。これらの機器は 1 つのバスに 15 台まで並列に接続できます。GPIB 機器は、以下の機能のうち 1 つ以上を備えています。

- トーカ
  バスにデータを送信するために指定された機器を「トーカ」と呼びます。GPIB バス上では、一台の機器のみがアクティブ・トーカとして動作します。
- リスナ
  バスのデータを受信するために指定された機器を「リスナ」と呼びます。アクティブなリスナ機器は GPIB バス上に複数存在できます。
- コントローラ
  トーカ、リスナを指定する機器を「コントローラ」と呼びます。GPIB バス上では一台の機器のみがアクティブ・コントローラとして動作します。これらのコントローラのうち、IFC、および REN のメッセージをコントロールできる機器を特に「システム・コントローラ」と呼びます。
  システム・コントローラは、GPIB バス上に一台だけ許されます。バス上に複数のコントローラがある場合、システム起動時にはシステム・コントローラがアクティブ・コントローラとなり、その他のコントローラ能力を持つ機器はアドレッサブル機器として動作します。
  その他のコントローラをアクティブ・コントローラにするには TakeControl(TCT) インタフェース・メッセージを用います。そのとき自分はノンアクティブ・コントローラとなります。
  コントローラはインタフェース・メッセージ、またはデバイス・メッセージを各測定器に送ってシステム全体をコントロールします。それぞれ以下の役目を果たします。
  ・インタフェース・メッセージ：GPIB バスをコントロールする
  ・デバイス・メッセージ　　：測定器をコントロールする

5.2.2 GPIB のセット・アップ

1. GPIB の接続

   以下に標準的な GPIB の接続を示します。GPIB コネクタは 2 本のねじでしっかり固定して、使用中にゆるむことがないように注意して下さい。

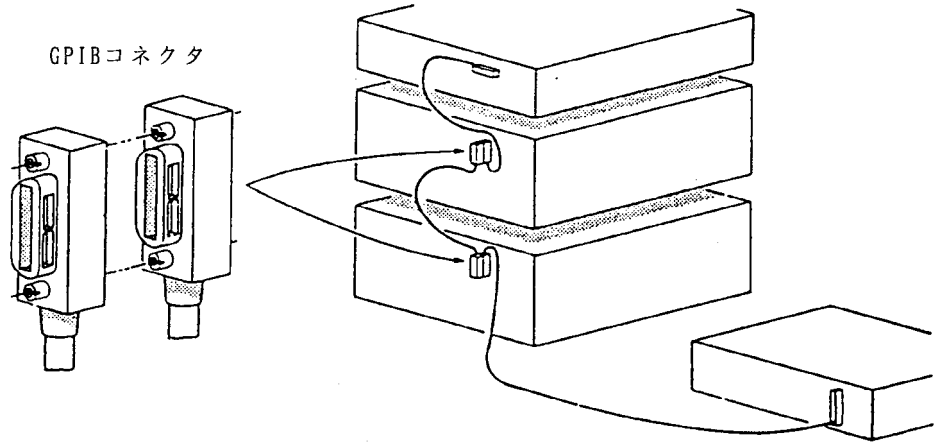


図 5-1　GPIB の接続

   GPIB インタフェースの使用時においては、以下のようなことに注意して下さい。

   - 1 つのバス・システムで使われる GPIB ケーブルの全ケーブル長は、2m × {接続される機器の数（GPIB コントローラも 1 つの機器として数える)} 以下です。
     また、ケーブルの全ケーブル長は 20m 以下とします。
   - 1 つのバス・システムに接続できる機器の数は、最高 15 台です。
   - ケーブル間の接続方法には制限はありません。ただし、1 台の機器上に 4 個以上の GPIB コネクタを重ねないで下さい。4 個以上重ねるとコネクタの取り付け部に過度の力が加わり、破損することがあります。
   - たとえば、5 台の機器から構成されるシステムで使用できる全ケーブル長は、10m 以下（5 台 × 2m/ 台 =10m）です。全ケーブル長が許容最大長を超えない範囲で、自由に分配することができます。ただし、10 台以上の機器を接続する場合は、何台かの機器を 2m 以下のケーブルで接続して、全ケーブル長が 20m を超えないようにする必要があります。

2. GPIB アドレスの設定

   GPIB アドレスは、System メニューの GPIB ADDRESS で設定します。設定方法は、「2.9.5 GPIB アドレスの設定 (GPIB ADDRESS)」を参照して下さい。

5.3 GPIB バスの機能

5.3.1 GPIB インタフェース機能

表 5-1 GPIB インタフェース機能

コード	説明
SH1	ソース・ハンドシェーク機能あり
AH1	アクセプタ・ハンドシェーク機能あり
T6	基本的トーカ機能、シリアル・ポール機能、リスナ指定によるトーカ解除機能
TE0	拡張トーカ機能なし
L4	基本的リスナ機能、トーカ指定によるリスナ解除機能
LE0	拡張リスナ機能なし
SR1	サービス・リクエスト機能あり
RL1	リモート機能、ローカル機能、ローカル・ロック・アウト機能
PP0	パラレル・ポール機能なし
DC1	デバイス・クリア機能
DT1	デバイス・トリガ機能
C0	システム・コントローラ機能なし
E1	オープン・コレクタ・バス・ドライバを使用

5.3.2 インタフェース・メッセージに対する応答

この節で説明するインタフェース・メッセージに対する本器の応答は、IEEE 規格 488.1-1987 および 488.2-1987 で定義されています。

インタフェース・メッセージの本器への送り方は、使用するコントローラの取扱説明書を参照して下さい。

5.3.2.1 インタフェース・クリア (IFC)

このメッセージは、本器へ直接信号線で送られてきます。

このメッセージによって本器は GPIB バスの動作を停止します。すべての入／出力を停止しますが、入出力バッファはクリアされません（クリアは DCL で実行される）。このとき本器がアクティブ・コントローラに指定されている場合、GPIB バスのコントロール権は解除され、システム・コントローラがコントロール権を得ます。

5.3.2.2 リモート・イネーブル (REN)

このメッセージは、本器へ直接信号線で送られてきます。

このメッセージが真のとき、本器がリスナに指定されるとリモート状態になります。

この状態は GTL を受けとるか、REN が偽になるか、LOCAL キーを押すまで続きます。

本器は、ローカル状態のとき、すべての受信データを無視します。

リモート状態のとき、LOCAL キーを除くすべてのキー入力を無視します。

ローカル・ロック・アウト状態（5.3.2.8 ローカル・ロック・アウト (LLO) を参照）のとき、すべてのキー入力を無視します。

5.3.2.3 シリアル・ポール・イネーブル (SPE)

本器はこのメッセージを外部から受信すると、シリアル・ポール・モードになります。

このモードでは、トーカに指定されると通常のメッセージではなくステータス・バイトを送信します。このモードはシリアル・ポール・ディセーブル (SPD) メッセージを受信するか、IFC メッセージを受信するまで続きます。

本器がサービス・リクエスト (SRQ) メッセージをコントローラに送信しているときには、応答データの bit6(RQS bit) が 1(TRUE) になります。送信が終了後、RQS bit は 0(FALSE) になります。サービス・リクエスト (SRQ) メッセージは、直接信号線で送ります。

5.3.2.4 グループ・エグゼキュート・トリガ (GET)

このメッセージは本器にトリガをかけ、本器は測定を始めます。

5.3.2.5 デバイス・クリア (DCL)

本器は DCL を受け取ったときに、以下のことを実行します。

- 入力バッファと出力バッファのクリア
- 構文解析部、実行コントロール部、応答データ生成部のリセット
- 次に実行するリモート・コマンドを妨げる全コマンドのキャンセル
- 他のパラメータを待つため一時停止されているコマンドのキャンセル
- *OPC と *OPC? のキャンセル

以下のことは実行しません。

- 本器に設定または格納されているデータの変更
- 正面パネル操作の中断
- 実行中の本器の動作への影響や中断
- MAV を除くステータス・バイトの変更（MAV は出力バッファのクリアの結果として 0 になる）

5.3.2.6 セレクテッド・デバイス・クリア (SDC)

DCL と同一の動作を行います。ただし、SDC は本器がリスナの場合だけ実行されます。

その他の場合は無視されます。

5.3.2.7 ゴー・トゥ・ローカル (GTL)

このメッセージは、本器をローカル状態にします。ローカル状態になると、正面パネル操作がすべて有効になります。

5.3.2.8 ローカル・ロック・アウト (LLO)

このメッセージは、本器をローカル・ロック・アウト状態にします。この状態で本器がリモート状態になると、正面パネル操作はすべて禁止されます（通常のリモート状態では、LOCAL キーで正面パネル操作ができる）。

このとき本器をローカル状態にする方法は、以下の 3 とおりあります。

- GTL メッセージを本器に送る
- REN メッセージを偽にする（このときローカル・ロック・アウト状態も解除される）
- 電源を再投入する

5.3.3 メッセージ交換プロトコル

本器は、コントローラやその他の機器から GPIB バスを通じてプログラム・メッセージを受け取り、応答データを発生します。プログラム・メッセージには、コマンド、クエリ（応答データを問い合わせるコマンドのことを特に「クエリ」と呼ぶ)、データが含まれています。それらのデータのやりとりには手順があります。この節ではその手順について説明します。

5.3.3.1 GPIB 各種バッファ

本器にはバッファが 3 つあります。

- 入力バッファ
  コマンド解析をするために一時的にデータを貯めておくバッファです。
  （1024 バイトの長さをもつ）
  入力バッファのクリア方法は、2 とおりあります。
  ・電源投入
  ・DCL または SDC の実行
- 出力バッファ
  コントローラからデータを読まれるまでデータを貯めておくバッファです。
  （1024 バイトの長さをもつ）
  出力バッファのクリア方法は、2 とおりあります。
  ・電源投入
  ・DCL または SDC の実行
- エラー・キュー
  IEEE488.2-1987 コマンド・モードでのみ存在します。
  これはリモート・コマンドのエラー・メッセージを蓄えておくキューで、深さは 10 です。リモート・コマンドの解析／実行でエラーが発生するたびに、メッセージがキューにつまれます。
  SYST:ERR コマンドで読み出すことができ、1 つ読み出すとキューから 1 つメッセージを削除します。
  エラー・キューのクリア方法は、2 とおりあります。
  ・電源投入
  ・*CLS の実行

5.3.3.2 IEEE488.2-1987 コマンド・モード

IEEE488.2-1987 コマンド・モードは、IEEE 規格 488.2-1987 に適合したメッセージ交換プロトコルに従ってメッセージの送受信を実行します。

このモードで、他のコントローラや機器がメッセージを本器から受信するときに特に重要な項目を、以下に示します。

- クエリの受信によって応答データを生成する
- クエリを実行した順にデータが生成される

パーサー

入力バッファから受信した順序通りにコマンド・メッセージを受け取り、構文解析を実行し、受け取ったコマンドがどんな内容の実行を行うのかを決定します。

コマンドの構文解析時にコマンドの木構造の追跡も行っています。
木構造のどの部分から解析すべきなのかを次のコマンドの解析のために覚えています。
この情報はパーサーがクリアされると木構造の頭まで戻ります。
パーサーのクリア方法は、4 とおりあります。

・電源投入

・DCL または SDC の受信

・’;’ の次の ’:’ の受信

・ターミネータまたは EOI の受信

応答データ生成

本器はパーサーがクエリを実行すると、その応答としてデータを出力バッファ上に生成します（つまりデータを出力するにはその直前に必ずクエリを送る必要がある）。

これはクエリで生成されるデータをコントローラがリードしなければデータがクリアされないことを意味します。

コントローラのリード以外でデータがクリアされる条件は 2 とおりあり、これらの状態は Query Error を発生します。

・Unterminated condition ； クエリをターミネート（ASCII の LF コードまたは GPIB の END メッセージ）せずにコントローラが応答データをリードしたか､クエリを送らずにコントローラが応答データをリードした場合

・Interrupted condition ； コントローラが応答データをリードする前に次のプログラム･メッセージを受け取った場合

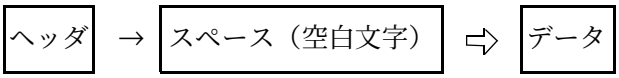
5.4 コマンド文法

5.4.1 IEEE488.2-1987 コマンド・モード

IEEE488.2-1987 コマンド・モードで入力する文字は、文字列データとブロック・データを除き英字の大文字・小文字の区別はありません。

5.4.1.1 コマンド文法

コマンド文法は、以下のフォーマットで定義されています。



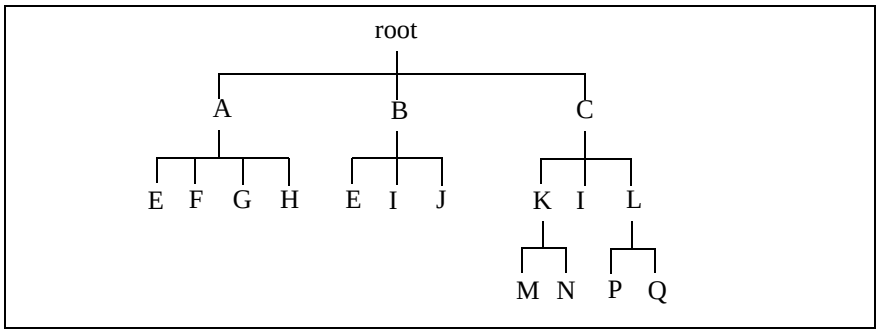
注　⇨ は繰り返しを意味します。

1. ヘッダ
   ヘッダは、コロン (:) で区切られた複数のニーモニックからなる階層構造を持ちます。4 文字以上からなるニーモニックは 4 文字（または 3 文字）の「ショート・フォーム」をもちます（省略しないニーモニックを「ロング・フォーム」と呼ぶ)。どちらのフォームをどのように組み合わせても構いません。
   ヘッダの直後に ? を付けるとクエリ・コマンドになります。
2. スペース（空白文字）
   1 文字分以上のスペースが必要です。スペース以外ではエラーとなります。
3. データ
   コマンドが複数のデータを必要とするときは、データをカンマ (,) で区切って複数並べます。カンマ (,) の前後にスペース（空白文字）を入れても構いません。
   データ・タイプの詳細については、5.4.1.2 データ・フォーマットを参照して下さい。
4. 複数のコマンドの記述
   IEEE488.2-1987 コマンド･モードでは複数のコマンドをセミコロン (;) で区切って 1 行で記述することが可能です。
   このようにコマンドを記述した場合には、ヘッダの持つ階層構造の中でカレント・パスを移動しながらコマンドを実行していきます。
5. カレント・パスの移動
   以下の規則に従ってカレント・パスは移動します。
   ・電源投入時　：カレント・パスは root にセットされる。
   ・ターミネータ　：カレント・パスは root にセットされる。
   ・コロン (:)　：カレント・パスをコマンド・ツリーの中で 1 階層下に移動するコロン (:) がコマンドの先頭の文字の場合、コロン (:) はカレント･パスを root にする。

・セミコロン (;) ： カレント・パスを変更しない。

・共通コマンド ： カレント・パスに関係なく実行できます。*RST コマンドを実行するとカレント・パスは root にセットされる（* 以下の例を参照)。

（例）以下のヘッダ構造とします。



このとき、以下のカレント・パス動作になります。

1. :A:E;:B:E

   2 つ目のコマンドの : はカレント・パスを root に移動するので、A:E と B:E はどちらも正しいコマンドです。

2. :A:E<END>B:E

   <END>（ターミネータ）はカレントパスを root に移動するので、A:E と B:E はどちらも正しいコマンドです。

3. :A:E;F;G;H

   ; はカレントパスを移動しないので、:A:E;F;G;H は結果的に A:E、A:F、A:G、A:H の 4 つのコマンドと等しくなります。

4. :C:I;K:N;M

   : がカレントパスを移動するので、K:N は :C: の階層から見ることになります。したがって K:N は C:K:N となります。また同時に、K:N は : を含むためカレント・パスを :C:K: に変更し、最後の M は C:K:M と解釈されます。

5. :A:E;*ESR 16

   共通コマンドはカレント・パスに関係ないので、*ESR 16 は正しく実行されます。

6 :A:E;*ESR 16;F;G;H

   共通コマンドはカレント・パスを変更しないので、3 つ目の F は 1 つ目の :A:E で設定されたカレント・パスの :A: で探されます。
   したがって、F は A:F、G は A:G、H は A:H になります。

以下の例では、文法エラーとなります。

1. :A:E;B:E

   A:E はカレント・パスを :A: に変更しています。
   したがって、B:E は :A: の階層で探されるが、B というニーモニックが見つからないのでエラーとなります。

2. :C:K:M;L:P

   :C:K:M はカレント・パスを :C:K: に変更しています。
   したがって、L:P は :C:K: で探されるが、L というニーモニックが見つからないのでエラーとなります。

5.4.1.2 データ・フォーマット

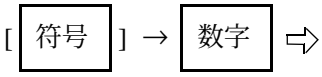
IEEE488.2-1987 コマンド・モードでは、この項で示すデータ・タイプをデータの入出力で使用します。

1. 数値データ

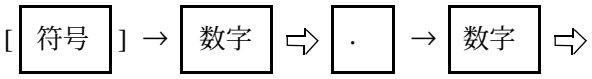
   数値データには以下の 3 つのフォーマットがあり、本器に対する数値の入力では、どれを用いても構いません（入力するデータの型に応じて四捨五入される）。

   また、コマンドによっては入力時に単位を付けられます。単位に関しては、後述 (5) を参照して下さい。

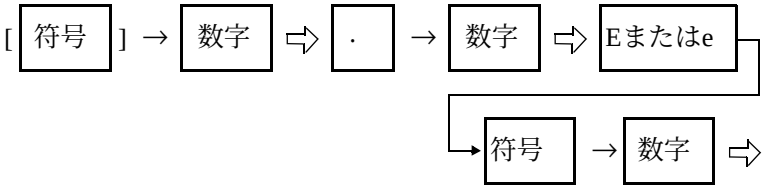
   - 整数型　　:NR1 フォーマット

   - 固定小数点型 :NR2 フォーマット

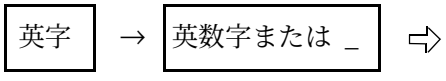
   - 浮動小数点型 :NR3 フォーマット

注　⇨ は繰り返しを意味します。
先頭の符合は省略可能です。

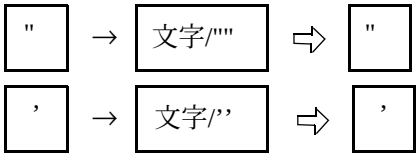
2. 文字データ

   文字データのフォーマットを以下に示します。

注　⇨ は繰り返しを意味します。

3. 文字列データ
   文字列データには、2 つのフォーマットがあります。

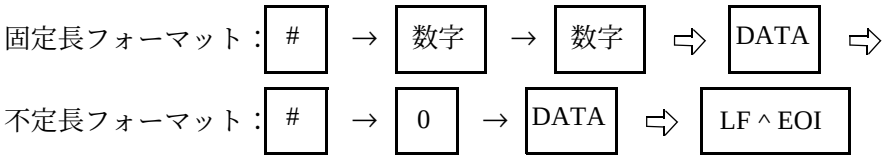


   文字列データ中では、ASCII 7bit コード文字として使用できます。

注　" で始まる文字列データ中では " を "" で表現しなければなりません。
' で始まる文字列データ中では ' を '' で表現しなければなりません。
⇨ は繰り返しを意味します。

   応答データが文字列データの場合、" で始まる文字列データを必ず出力します。

4. ブロック・データ
   ブロック・データには、2 つのフォーマットがあります。本器への入力時には、どちらのフォーマットを用いても構いません。



注　⇨ は繰り返しを意味します。

   固定長のフォーマットでは、# の後の 1 文字の数字でその後に続くデータのバイト数の桁数を表します。0 は使えません（不定長になる）。

   （例）#3128<data byte> というブロックデータの場合
   # の後の 3 がその後に続く文字列 (128) の桁数を表し、128 はその後に続く <data byte> のバイト数を表します。

   また、バイナリ・データ・フォーマットは、単精度 (32bit) と倍精度 (64bit) を選択することができます。

   単精度フォーマット : 仮数部の符号 (bit31)、指数部 (bit30 ～ 23)、仮数部 (bit22 ～ 0)

   倍精度フォーマット : 仮数部の符号 (bit63)、指数部 (bit62 ～ 52)、仮数部 (bit51 ～ 0)

5. 単位

単位は数値の後に続く接尾語です。また、単位にはサフィックスを接頭語として使用できます。

使用可能なサフィックスと単位の一覧表を以下に示します。

表 5-2　使用可能なサフィックスと単位

サフィックス		単位	使用可能なコマンド例
1E18	EX	M	:DISPlay[:WINDow]:TRAce:X[SCALe]:CENTer :DISPlay[:WINDow]:TRAce:X[SCALe]:SPAN {:MEASure|:FETCh}[:SCALar]:LENGth:COHerency:PEAK:RANGe:STARt {:MEASure|:FETCh}[:SCALar]:LENGth:COHerency:PEAK:RANGe:STOP
1E15	PE		
1E12	T		
1E9	G	DB	:CALCulate3:BANDwidth:XDB :CALCulate3:BANDwidth:YDB
1E6	MA		
1E3	K	DBM	:SENSe:POWer:RLEVel :DISPlay[WINDow]:TRACe:Y[:SCALe]:RLEVel :DISPlay[WINDow]:MARKer3:POWer :DISPlay[WINDow]:MARKer4:POWer
1E-3	M*		
1E-6	U		
1E-9	N	W	:DISPlay[WINDow]:MARKer3:POWer :DISPlay[WINDow]:MARKer4:POWer
1E-12	P		
1E-15	F		
1E-18	A		

*:　単位が HZ の場合、サフィックスは 1E6（MA と同等）として動作します。

5.5 ステータス・バイト

本器では IEEE 規格 488.2-1987 に適合した階層化されたステータス・レジスタ構造をもち、機器の様々な状態をコントローラへ送信できます。ここではこのステータス・バイトの動作モデルと、イベントの割当を説明します。

1. ステータス・レジスタ

   本器は、IEEE 規格 488.2-1987 で定義されたステータス・レジスタのモデルを採用し、コンディション・レジスタ、イベント・レジスタ、イネーブル・レジスタから構成されています。

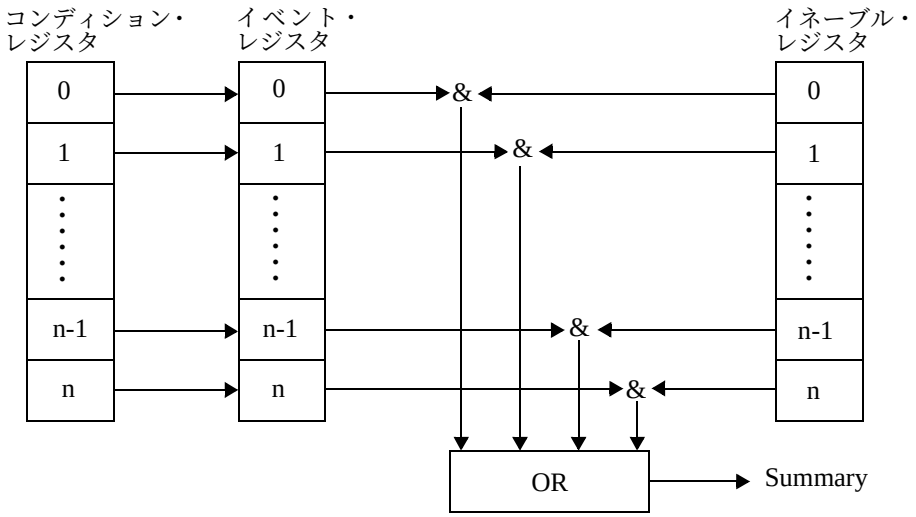


図 5-2　ステータス・レジスタの構成

   a. コンディション・レジスタ

      コンディションレジスタは、機器のステータスを常に監視しています。つまり、このレジスタには常に最新の機器のステータスが保持されています。
      ただし、コンディション・レジスタは内部情報として保持しているので、データの読み書きはできません。

   b. イベント・レジスタ

      イベント・レジスタは、コンディション・レジスタからのステータスをラッチして保持します（変化を保持する場合もある）。
      このレジスタがセットされると、クエリで読み出されるか、*CLS でクリアされるまでセットされたままです。
      イベント・レジスタにデータを書き込むことはできません。

   c. イネーブル・レジスタ

      イネーブル・レジスタは、イベント・レジスタのどのビットを有効なステータスとしてサマリを生成するのか指定します。イネーブル・レジスタはイベント・レジスタと AND をとられ、その結果の OR がサマリとして生成されます。サマリはステータス・バイト・レジスタに書き込まれます。
      イネーブル・レジスタはデータを書き込めます。

本器のステータス・レジスタは、以下の 5 種類があります。

- ステータス・バイト・レジスタ
- スタンダード・イベント・レジスタ
- スタンダード・オペレーション・ステータス・レジスタ
- クエスチョナブル・ステータス・レジスタ
- デバイス・ステータス・レジスタ

本器のステータス・レジスタの配置を図 5-3 に示します。
ステータス・レジスタの詳細を図 5-4 に示します。

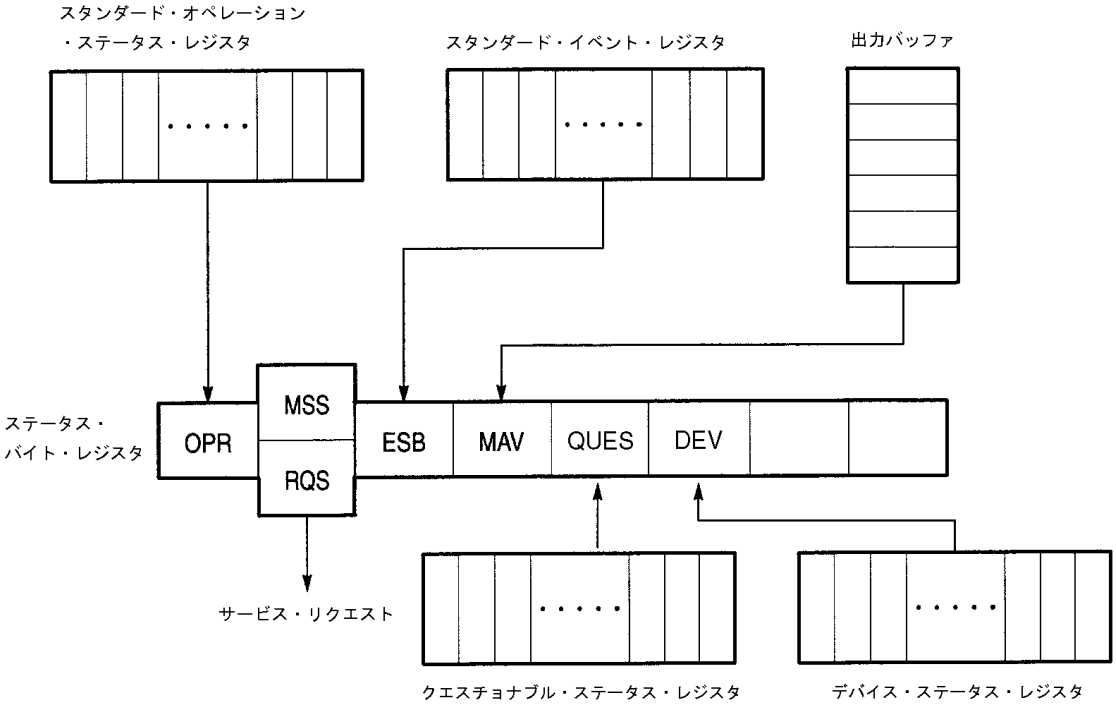


図 5-3　ステータス・レジスタの配置

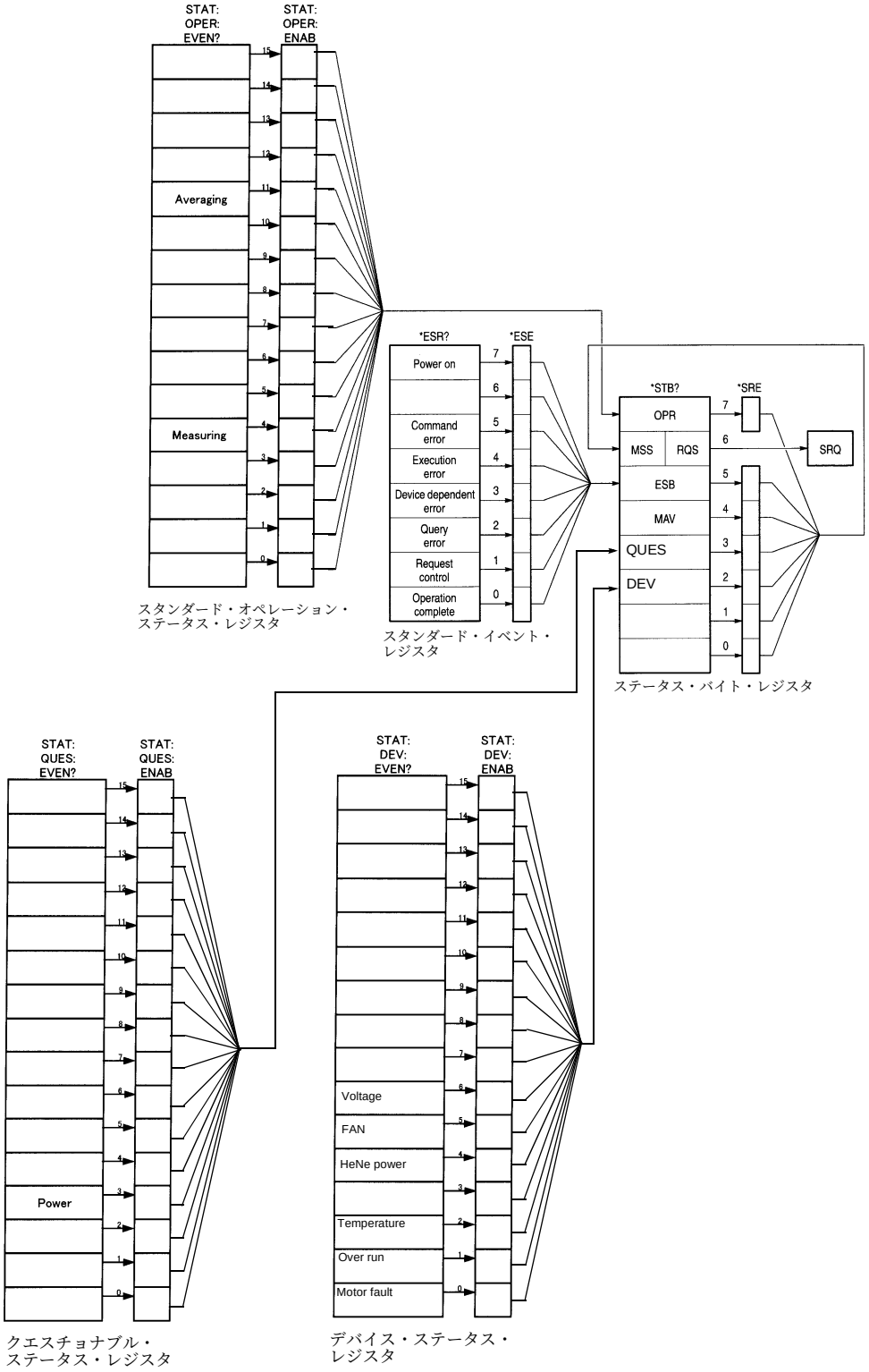


図 5-4　ステータス・レジスタの詳細

2. イベント・イネーブル・レジスタ
   各イベント・レジスタには、どのビットを有効にするかを決めるイネーブル・レジスタがあります。
   - サービス・リクエスト・イネーブル・レジスタ のセット : *SRE
   - スタンダード・イベント・ステータス・イネーブル・レジスタのセット : *ESE
   - オペレーション・ステータス・イネーブル・レジスタのセット : STAT:OPER:ENAB
   - クエスチョナブル・ステータス・イネーブル・レジスタのセット : STAT:QUES:ENAB
   - デバイス・ステータス・イネーブル・レジスタのセット : STAT:DEV:ENAB
3. スタンダード・オペレーション・ステータス・レジスタ
   スタンダード・オペレーション・ステータスのイベント・レジスタの割り当てを、以下に示します。

表 5-3　スタンダード・オペレーション・ステータス・レジスタの割り当て

bit	機能定義	説明
15 〜 12		常に 0
11	Averaging	アベレージ終了時に 1 にセットされる。
10 〜 5		常に 0
4	Measuring	測定終了時に 1 にセットされる。
3 〜 0		常に 0

4. クエスチョナブル・ステータス・レジスタ
   クエスチョナブル・ステータスのイベント・レジスタの割り当てを、以下に示します。

表 5-4　クエスチョナブル・ステータス・レジスタの割り当て

bit	機能定義	説明
15 〜 4		常に 0
3	Power	過剰レベルの信号が入力されたときに 1 にセットされる。
2 〜 0		常に 0

5. デバイス・ステータス・レジスタ
デバイス・ステータスのイベント・レジスタの割り当てを、以下に示します。

表 5-5　デバイス・ステータス・レジスタの割り当て

bit	機能定義	説明
15 〜 7		常に 0
6	Voltage	電圧異常時に 1 にセットされる。
5	FAN	ファン異常時に 1 にセットされる。
4	HeNe power	He-Ne レーザのパワー不足時に 1 にセットされる。
3		常に 0
2	Temperature	温度異常上昇時に 1 にセットされる。
1	Over run	可動鏡の動作異常時に 1 にセットされる。
0	Motor fault	可動鏡およびその制御部の異常時に 1 にセットされる。

6. ステータス・バイト・レジスタ
ステータス・バイト・レジスタは、ステータス・レジスタからの情報を要約しています。また、このステータス・バイト・レジスタのサマリがサービス・リクエストとしてコントローラに送信されます。そのため、ステータス・バイト・レジスタは、ステータス・レジスタ構造とは若干違った動作を行います。ここではステータス・バイト・レジスタに関して説明をします。
ステータス・バイト・レジスタの構造を、図 5-5 に示します。

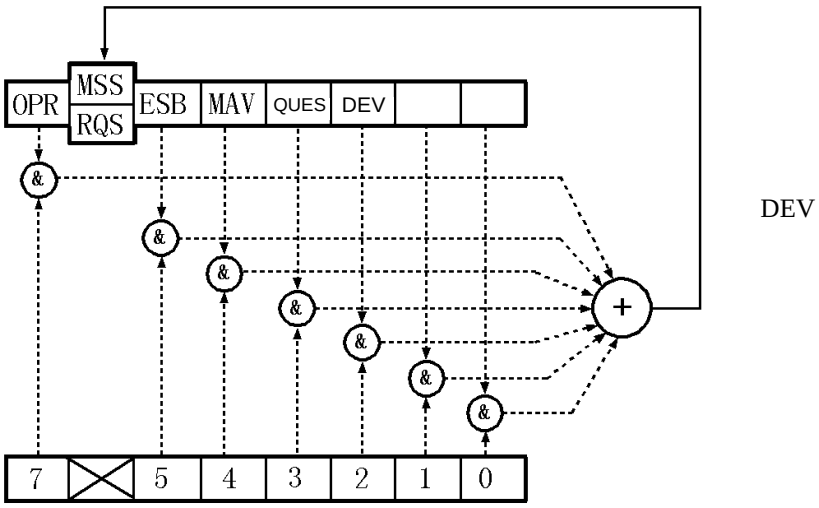


図 5-5　ステータス・バイト・レジスタの構造

このステータス･バイト･レジスタは､以下の 3 点を除くとステータス･レジスタに従います。

- ステータス・バイト・レジスタのサマリが、ステータス・バイト・レジスタの bit6 に書き込まれます。
- イネーブル・レジスタの bit6 は、常に有効で変更できません。
- ステータス・バイト・レジスタの bit6 (MSS) が、サービス・リクエスト要求の RQS を書き込みます。

このレジスタが､コントローラからのシリアル･ポールに対して応答します。シリアル･ポールに対して応答するときには、ステータス・バイト・レジスタの bit0 ～ 5、bit7 および RQS が読み出され､その後に RQS は 0 にリセットされます。その他のビットはそれぞれの要因が 0 になるまでクリアされません。
ステータス・バイト・レジスタ、RQS、MSS は、“*CLS” を実行するとクリアできます。それにともなって、SRQ ラインも偽になります。
ステータス・バイト・レジスタの各ビットの意味を、以下に示します。

表 5-6　ステータス・バイト・レジスタの意味

bit	機能定義	説明
7	OPR	OPR は、スタンダード・オペレーション・ステータス・レジスタのサマリである。
6	MSS	RQS は、ステータス・バイト・レジスタの MSS が 1 になったとき TRUE になるが、その MSS はすべてのステータス・データ構造のサマリ・ビットになっている。 MSS は、シリアル・ポールでは読めない（ただし、RQS が 1 のときは MSS が 1 であることがわかる）。 MSS を読むには、共通コマンド *STB? を用いる。 *STB? ではステータス・バイト・レジスタの bit0 ～ 5 、bit7 および MSS が読み出される。 この場合ステータス・バイト・レジスタと MSS はクリアされない。 MSS は、ステータス・レジスタ構造のすべてのマスクされていない要因がクリアされるまで 0 にならない。
5	ESB	ESB は、スタンダード・イベント・レジスタのサマリである。
4	MAV	出力バッファの要約ビット 出力バッファに出力データがある間 1 になり、データが読み出されると 0 になる。
3	QUES	QUES は、クエスチョナブル・ステータス・レジスタのサマリである。
2	DEV	DEV は、デバイス・ステータス・レジスタのサマリである。
1 ～ 0		常に 0

7. スタンダード・イベント・レジスタ
   スタンダード・イベント・レジスタの割り当てを、以下に示します。

表 5-7　スタンダード・イベント・レジスタの割り当て

bit	機能定義	説明
7	Power on	電源投入で 1 になる。
6		常に 0
5	Command Error	パーサーが文法エラーを見つけたときに 1 にセットされる。
4	Execution Error	GPIB コマンドとして受け取った命令の実行を何らかの理由（パラメータが範囲外など）で失敗すると 1 にセットされる。
3	Device Dependent Error	Command Error 、 Execution Error 、 Query Error 以外のエラーが発生したとき 1 にセットされる。
2	Query Error	コントローラが本器からデータを読み出そうとしたときに、データが存在しない、またはデータが消失していると 1 にセットされる。
1	Request Control	本器がアクティブ・コントローラになる必要があるときに 1 にセットされる。
0	Operation Complete	*OPC コマンドを受け取った後、かつ本器に実行しているコマンドがなくなると、1 にセットされる。

5.6 コマンド・リファレンス

この章では、本器のすべてのリモート・コマンドの文法（コマンド文法、クエリ文法、または両方）、応答データ・フォーマット（クエリの存在するとき）、およびコマンドの詳細の説明をします。

注　コマンドを参照する場合、コマンド・ニーモニックの一部を省略可能なことを考慮に入れて下さい。

（例）以下の 2 つのコマンドは、表記は違いますが同じものです。

:DISP:TRAC SPEC
:DISP:WIND:TRAC SPEC

コマンドの詳細は、以下のようなサブシステムごとに分かれています。

共通コマンド	: すべての測定器で同じ動作をするコマンドです。
MEASURE コマンド	: 測定の開始／停止を実行するコマンドです。
RANGE コマンド	: 波長帯域の設定をするコマンドです。
SETUP コマンド	: 基本的な測定条件を設定するコマンドです。
APPLICATION コマンド	: 測定アプリケーション・コマンドです。
SCALE コマンド	: 表示画面の条件を設定するコマンドです。
CURSOR コマンド	: カーソル関係のコマンドです。
SYSTEM コマンド	: システム関係のコマンドです。
GPIB 機能コマンド	: GPIB 専用のコマンドです。

5.6.1 コマンド記述のフォーマットの説明

以降の節で IEEE488.2-1987 のコマンド・モードの詳細を説明をします。

以下の注意事項を参照して下さい。

注意

1. **コマンドと応答データ・フォーマットは、以下の記号を用いて記述します。**
   記号 <> : **文法の構成要素を示すその内容は、その後に記述される。**
   記号 | : **複数の中から一つを選択することを示す。**
   （例）A | B | C**これは** A **、** B **または** C **という意味。**
   記号 [ ] : **囲まれた項目は、オプション（省略可能）であることを示す。**
   記号 { } : **囲まれた項目は、グループを表し、**{ } **の中で** | **で区切られた複数の項目の** 1 **つを選択することを示す。**
2. 4 **文字以上のニーモニックはショート・フォームをもちます。本文中では大文字で記述した部分がショート・フォームになります。**
   （例）DISPlay:MARKer:MODE
   ショート・フォーム :DISP, MARK
   ロング・フォーム :DISPLAY, MARKER
   MODE **は** 4 **文字なのでショート・フォームとロング・フォームの区別はありません。**
3. **クエリは、コマンドのヘッダに** ? **をつけます。パラメータを必要とするクエリは、クエリのフォーマットも記述します。**
4. **この章で共通に用いているパラメータの書式を以下に示します。**

<ch>	: **チャンネル番号**	1 - 300, **省略** = **アクティブ・チャンネル**
<bool>	: **真偽値**	0|1|OFF|ON **のいずれか** 0 **と** OFF**、**1 **と** ON **が対応**
<int>	: **整数値**	
<real>	: **実数値**	
<str>	: " **文字列** "	
<block>	: **ブロックデータ**	
-	: **パラメータ指定なし**	
×	: **使用不可**	

5.6.2 共通コマンド

1. *CLS
   - 機能　ステータス・バイトと関連データのクリア
   - コマンドとクエリの存在　Command
   - コマンド　*CLS
   - 説明　*CLS はステータス・データ構造をクリアし、強制的に *OPC と *OPC? をキャンセルします。また、エラー・キューもクリアします。しかし、このコマンド自身は出力バッファをクリアしないので、出力データがある場合 MAV ビットはクリアされません。ただし、行の最初にこのコマンドを実行するとデータがクリアされるので、MAV を含めてすべてのステータスがクリアされます。
     *CLS は、エラー・キューのクリアも実行します。

2. *DDT

- 機能　GET に対するマクロ定義
- コマンドとクエリの存在　Command/Query
- コマンド　*DDT <block>
- パラメータ　<block>
- 応答形式　<block>
- 説明　*DDT は *TRG、または GET インタフェース・メッセージが受信されたときに実行するコマンド・シーケンスを定義します。つまり、*TRG の動作を <block> データ中に記述された一連のコマンドと置き換えます。定義できるシーケンスの長さは 255 文字以内です。

  *DDT で 0 の長さのブロック・データ (#10) を定義すると、*TRG および GET インタフェース・メッセージで何も実行しないことを定義することになります。また、*RST の実行でマクロをキャンセルします。

  クエリに対する応答は、ブロック・データで応答します。マクロが未定義の状態で *DDT? を実行すると、0 の長さのブロックデータ (#10) が返ります。
- 注意　この定義中に *TRG は用いないで下さい。*DDT で定義中に *TRG を用いるとトリガではなく、*DDT で設定したシーケンスを呼び出し、無限ループとなります（実際にはネスティングの制限にかかり、マクロ・エラーになります）。
- 例　*DDT #212INIT:CONT ON のとき

  *TRG → INIT:CONT ON と置き換えます。

3. *DMC

- 機能　マクロ定義
- コマンドとクエリの存在　Command
- コマンド　*DMC <str>,<block>
- パラメータ　<str>
  <block>
- 説明　*DMC は <str> で指定されたマクロ・ラベルにコマンド・シーケンスを定義します。この定義で本器は <str> を受信したときに <block> の本体を受信したのと同じ動作を実行するようになります（ただし、*EMC 1 でなければならない）。
  このマクロ・ラベルには階層コマンドも使用できます。また、あらかじめ定義されているコマンドにマクロを上書きすることもできます（ただし、共通コマンドには上書きできない）。このときは *EMC 1 でマクロをイネーブルにするとマクロで置き換えた一連のコマンドを、*EMC 0 でディセーブルにすると本来の動作を行います。*DMC で定義したマクロの削除は *PMC を用いて下さい。一度登録したマクロは *PMC でクリアされるまで再登録できません。
  マクロの本体はコマンドの文法に従って記述して下さい。マクロ・コマンドに与えたパラメータは $1 ～ $9 で 9 個まで表現できます。数字はマクロコマンドに続くパラメータが 1、次が 2 と進みます。また、マクロ定義にマクロを含むことができます。最大 9 レベルまでのネスティングをサポートしています。登録可能な新規マクロは最大 30 です（条件によって変化する）。
  *PMC, *GMC?, *LMC?, *EMC も参照して下さい。
- 例　*DMC "LIMIT",
  #241:MEAS:SCAL:LENG:COH:RANG:STAR $1;STOP $2; のとき
  LIMIT 0.002, 0.010 →
  :MEAS:SCAL:LENG:COH:RANG:STAR 0.002;STOP 0.010;
  と置き換えます。

4. *EMC
   - 機能　マクロの実行許可
   - コマンドとクエリの存在　Command/Query
   - コマンド　*EMC <int>
   - パラメータ　<int>
   - 応答形式　0 | 1
   - 説明　マクロの実行の許可 (1) または禁止 (0) をします。このコマンドはマクロの定義内容に影響を与えません。
     このコマンドはマクロで上書きされた本来のコマンドを実行するなどのために使用します。
     *RST でマクロの実行は禁止されます。
     *DMC, *PMC, *GMC?, *LMC? も参照して下さい。
5. *ESE
   - 機能　スタンダード・イベント・ステータス・イネーブル・レジスタの設定
   - コマンドとクエリの存在　Command/Query
   - コマンド　*ESE <int>
   - パラメータ　<int>
   - 応答形式　NR1（整数値）
   - 説明　スタンダード・イベント・ステータス・レジスタのイネーブル・レジスタを設定します。このレジスタの 1 に設定された bit に対応するスタンダード・イベント・ステータス・レジスタが、有効ビットとしてステータス・バイト・レジスタに反映します。
     詳細はステータス・データ構造の説明を参照して下さい。
     *ESR? も参照して下さい。
   - 例　Operation Complete(bit0) と Device Dependent Error(bit3) をイネーブルにセットするとき
     $2^3 + 2^0 = 8 + 1 = 9$ と計算し、*ESE 9 とセットします。

6. *ESR?

- 機能　スタンダード･イベント･ステータス･レジスタの読み出し
- コマンドとクエリの存在　Query
- クエリ　*ESR?
- 応答形式　NR1（整数値）
- 説明　スタンダード・イベント・ステータス・レジスタの値を読み出します。
  スタンダード・イベント・ステータス・レジスタは読み出すとクリアされ､対応するステータスバイトのビット (bit5) もクリアされます。
  詳細はステータス・データ構造の説明を参照して下さい。

表 5-8　スタンダード・イベント・レジスタの割り当て

bit		説明
7	Power on	電源 ON で 1 になる
6		常に 0
5	Command Error	パーサーが文法エラーを見つけたときに 1 にセットされる
4	Execution Error	GPIB コマンドとして受け取った命令の実行を何らかの理由（パラメータが範囲外など）で失敗すると 1 にセットされる
3	Device Dependent Error	Command Error、Execution Error、QuerError 以外のエラーが発生したとき 1 にセットされる
2	Query Error	コントローラが本器からデータを読み出そうとしたときに、データが存在しない、またはデータが消失していると 1 にセットされる
1	Request Control	本器がアクティブ・コントローラになる必要があるときに 1 にセットされる
0	Operation Complete	*OPC コマンドを受け取った後で、かつ本器が実行しているコマンドがなくなると 1 にセットされる

7. *GMC?
   - 機能　マクロ定義の問い合わせ
   - コマンドとクエリの存在　Query
   - クエリ　*GMC? <name>
   - パラメータ　<name>
   - 応答形式　<block>
   - 説明　<name> で指定したマクロの定義を読み出します。
     未定義の <name> マクロを *GMC? で読み出すと、0 の長さのブロック・データ (#10) が返ります。
     *DMC, *PMC, *LMC?, *EMC も参照して下さい。
8. *IDN?
   - 機能　機器の問い合わせ
   - コマンドとクエリの存在　Query
   - クエリ　*IDN?
   - 応答形式　"<manufacturer>,<model>,<serial number>,<firmware level>"
     <manufacturer> = ADC Corp.
     <model> = 機種名
     <serial number> = シリアル番号
     <firmware level> = システム・バージョン
   - 説明　本器の識別情報を取り出します。上記の応答形式の項目で記述している 4 項目を文字列形式で出力します。
9. *LMC?
   - 機能　すべてのマクロ定義の読み出し
   - コマンドとクエリの存在　Query
   - クエリ　*LMC?
   - 応答形式　"<macro label>"[,"<macro label>"...]
     <macro label> = マクロ・ヘッダ
   - 説明　すべてのマクロ・ヘッダを文字列形式で応答します。複数のマクロが定義されているときは,で区切って並べます。定義されているマクロがない場合は、0 文字長の文字列 ("") で応答します。
     *DMC, *PMC, *GMC?, *EMC も参照して下さい。

10. *OPC

- 機能　実行中のすべての動作の終了の通知
- コマンドとクエリの存在　Command/Query
- コマンド　*OPC
- 応答形式　1
- 説明　*OPC は現在実行中のすべてのコマンドが終了したときに標準イベント・ステータス・レジスタの‘ Operation Complete’bit を 1 に設定します。“ 現在実行中のすべてのコマンド " が終了する前に次のコマンドを受けとると、そのコマンド実行の終了も待ちます。つまり、*OPC を受けとった後に本器が何も実行していない状態になったときにステータス・レジスタの設定をします。

  *OPC? は上記の *OPC で設定する‘ Operation Complete’bit の代わりに出力バッファに 1 を書き込みます。つまり、コントローラが本器からの応答を受けとるタイミングでコマンド終了のタイミングをとれます。

  *OPC、*OPC? ともに DCL インタフェース・メッセージ、*CLS、および *RST で解除されます。

  *WAI も参照して下さい。

11. *PMC

- 機能　すべてのマクロ定義の削除
- コマンドとクエリの存在　Command
- コマンド　*PMC
- 説明　*PMC はすべてのマクロ定義を削除します。このコマンドで本器のメモリからすべてのマクロ・ヘッダとマクロ本体が削除され、新しいマクロの登録が可能になります。

  *DDT, *DMC, *GMC?, *LMC?, *EMC も参照して下さい。

12. *RCL

- 機能　機器の設定のリコール
- コマンドとクエリの存在　Command

  コマンド　*RCL{<int> | POFF}

  パラメータ　<int> = レジスタ番号 (0 ～ 9999)
  POFF = 前回のパワーオフ時の設定
- 説明　本器の設定条件を指定した内部レジスタから呼び出します。
  レジスタ番号 0 または POFF（または RECLPOFF）は前回のパワーオフ時の設定値を呼び出します。

13. *RST

- 機能　機器のリセット
- コマンドとクエリの存在　Command
- コマンド　*RST
- 説明　*RST は本器のリセットを実行します。実際には以下のことを実行します。
  1. 本器の設定を初期状態にする。
  2. *DDT で定義されるマクロを初期状態にする。
  3. マクロを無効にする（*EMC 0 と同じ）。
  4. *OPC、*OPC? を無効にする。

  以下への影響はありません。
  1. GPIB バスの状態
  2. GPIB アドレス
  3. 出力バッファ
  4. ステータスデータ構造
  5. *DMC で定義するマクロ
  6. デバイスの校正データ

  SYSTem:PRESet (IP) も参照して下さい。

14. *SAV

- 機能　機器の設定のセーブ
- コマンドとクエリの存在　Command

  コマンド　*SAV <int>

  パラメータ　<int> = レジスタ番号 (1 ～ 9999)
- 説明　本器の設定条件を指定した番号の内部レジスタに記憶します。

  セーブ・レジスタは実行されると各データをファイル化し内蔵ハード・ディスク（C ドライブ）に保存します。C ドライブのサイズを超えてデータを保存することはできません。メモリ容量を超えた場合、保存は実行されません。保存されている他のデータを消去してから再度保存して下さい。

15. *SRE

- 機能　サービス・リクエスト・イネーブル・レジスタの設定
- コマンドとクエリの存在　Command/Query
- コマンド　*SRE <int>
- パラメータ　<int>
- 応答形式　NR1 （整数値）
- 説明　サービス・リクエスト・イネーブル・レジスタを設定します。このレジスタの 1 に設定された bit に対応するステータス・バイト・レジスタが有効ビットとして MSS に反映します。

  クエリ時の応答データ bit6 は、常に 0 となります。

  詳細はステータス・データ構造の説明を参照して下さい。*STB? も参照して下さい。
- 例　OPR(bit7)、ESB(bit5) および MAV(bit4) をイネーブルにセットするとき

  $2^7 + 2^5 + 2^4 = 128 + 32 + 16 = 176$

  と計算し、*SRE 176 とセットします。

16. *STB?

- 機能　ステータス・バイト・レジスタの読み出し
- コマンドとクエリの存在　Query
- クエリ　*STB?
- 応答形式　NR1（整数値）
- 説明　ステータス・バイト・レジスタの内容を読み出します。
  ここで読み出されるリクエストの要約ビットは MSS です。
  このレジスタと MSS は読み出されてもクリアされません。
  詳細はステータス・データ構造の説明を参照して下さい。

表 5-9　ステータス・バイト・レジスタの割り当て

bit		説明
7	OPR	OPR は、スタンダード・オペレーション・ステータス・レジスタのサマリである
6	MSS	RQS はステータス・バイト・レジスタの MSS が 1 になったときに TRUE となるが、その MSS はすべてのステータス・データ構造のサマリ・ビットになっている
		MSS は、サービス・リクエストでは読めない（ただし、RQS が 1 のときは MSS が 1 であることがわかる）
		MSS を読むには、共通コマンドの *STB? を用いる *STB? ではステータス・バイト・レジスタの bit0 ～ 5、bit7 および MSS が読み出される この場合ステータス・バイト・レジスタと MSS はクリアされない
		MSS は、ステータス・レジスタ構造のすべてのマスクされていない要因がクリアされるまで 0 にならない
5	ESB	ESB は、スタンダード・イベント・レジスタのサマリである
4	MAV	MAV は出力バッファの要約ビット
		出力バッファに出力データがある間 1 になり、データが読み出されると 0 になる
3	QUES	QUES は、クエスチョナブル・ステータス・レジスタのサマリである
2	DEV	DEV は、デバイス・ステータス・レジスタのサマリである
1 ～ 0		常に 0

17. *TRG
    - 機能　機器にトリガをかける
    - コマンドとクエリの存在　Command
    - コマンド　*TRG
    - 説明　*TRG は機器にトリガをかけます。*TRG を受けとると、本器は測定を開始します。
      *TRG、GET インタフェース・メッセージともに入力バッファにつまれ、入力順に処理されます。
18. *TST?
    - 機能　セルフテストの結果の問い合わせ
    - コマンドとクエリの存在　Query
    - クエリ　*TST?
    - 応答形式　0 | エラー・コード
    - 説明　*TST? は本器にセルフテストを実行させ、その結果を応答します。0 の応答はセルフテストの成功を意味し、それ以外の応答はエラー・コードを意味します。
19. *WAI
    - 機能　実行中のすべての動作の終了を待つ
    - コマンドとクエリの存在　Command
    - コマンド　*WAI
    - 説明　*WAI は現在実行中のすべてのコマンドが終了するのを待ちます。
      このコマンドを実行すると、これ以降のすべてのコマンドは現在実行中のコマンドの終了まで遅延されます。
      *WAI は DCL インタフェース・メッセージでキャンセルされます。

5.6.3 MEASURE コマンド

表 5-10　MEASURE コマンド

機能	コマンド	パラメータ	クエリ
測定スタート	:INITiate:IMMediate	-	×
測定ストップ	:ABORt	-	×
連続測定 ON/OFF	:INITiate:CONTinuous	ON|OFF	ON|OFF
測定デバイス (LD/LED)	:SENSe:CORRection:DEVice	NARRow|BROad	NARR|BRO

5.6.4 RANGE コマンド

表 5-11　RANGE コマンド

機能	コマンド	パラメータ	クエリ
CW 光測定レンジ	:SENSe:RANGe	0|1|2|3=SHORT|MIDDLE1|MIDDLE2|LONG	0|1|2|3=SHORT|MIDDLE1|MIDDLE2|LONG
CW 光測定レンジ・オート	:SENSe:RANGe:AUTO	ON|OFF	ON|OFF
パルス光測定レンジ	:SENSe:RANGe:PULSe	0|1=NARROW|BROAD	0|1=NARROW|BROAD
測定モード（パルス光／ CW 光）	:SENSe:MODE	0|1=CW|PULSE	<int>

5.6.5 SETUP コマンド

表 5-12 SETUP コマンド

機能	コマンド	パラメータ	クエリ
中心波長および中心光路長	:DISPlay[:WINDow]:TRACe:X[:SCALe]:CENTer	<real>	<real>
波長スパンおよび光路差	:DISPlay[:WINDow]:TRACe:X[:SCALe]:SPAN	<real>	<real>
ピーク・センタ	:DISPlay[:WINDow]:TRACe:X[:SCALe]:CENTer:PEAK	-	×
カーソル・センタ	:DISPlay[:WINDow]:TRACe:X[:SCALe]:CENTer:MARKer	-	×
カーソル・スパン	:DISPlay[:WINDow]:TRACe:X[:SCALe]:SPAN:MARKer	-	×
測定分解能	:SENSe:SWEep:POINts	NORMal|HIGH	NORM|HIGH
スタート波長およびスタート光路長	:DISPlay[:WINDow]:TRACe:X[:SCALe]:LEFT	<real>	<real>
ストップ波長およびストップ光路長	:DISPlay[:WINDow]:TRACe:X[:SCALe]:RIGH	<real>	<real>
X 軸設定選択	:DISPlay[:WINDow]:TRACe:X[:SCALe]:ACTIve	SPECtrum|COHerence	SPEC|COH
リファレンス・レベル	:SENSe:POWer:RLEVel	<real>	<real>
オート・リファレンス・レベル	:SENSe:POWer:RLEVel:AUTO	ON|OFF	ON|OFF
アベレージ ON/OFF	:CALCulate1:CAVErage[:STATe]	ON|OFF	ON|OFF
アベレージ回数	:CALCulate1:CAVErage:COUNt	<int>	<int>
アベレージ選択	:CALCulate1:CAVErage:TYPE	SN|POWer	SN|POW
スムージング機能	:CALCulate2:SMOothing[:STATe]	ON|OFF	ON|OFF
スムージング・ポイント数	:CALCulate2:SMOothing:APERture	5|7|9|11|13	5|7|9|11|13
スムージング・スレッショルド	:CALCulate2:SMOothing:THReshold	<real>	<real>
波形表示データ (*1)			
波形データ数	:DISPlay[:WINDow]:TRACe:POINts?	SPECtrum|COHerence	<int>
X 軸波形データ	:DISPlay[:WINDow]:TRACe:X:DATA?	SPECtrum|COHerence	*FORMAT
Y 軸波形データ	:DISPlay[:WINDow]:TRACe[:Y]:DATA?	SPECtrum|COHerence	*FORMAT
波形フルデータ数	:DISPlay[:WINDow]:TRACe:FULL:POINts?	SPECtrum|COHerence	<int>
X 軸波形フルデータ	:DISPlay[:WINDow]:TRACe:X:FULL:DATA?	SPECtrum|COHerence	*FORMAT
Y 軸波形フルデータ	:DISPlay[:WINDow]:TRACe[:Y]:FULL:DATA?	SPECtrum|COHerence	*FORMAT
ピーク・サーチ・データ	{:MEASure|FETCh}[:SCALar]:PEAK?	SPECtrum|COHerence	“<real>,<real>= 波長、レベル（SPECtrum 指定）or <real>,<real>,<real>,<real>= α 距離、α レベル、β 距離、β レベル (COHerency)”
スペクトラム 2nd ピーク・データ	{:MEASure|FETCh}[:SCALar}:PEAK:SEcond?	×	“<real>,<real>= 波長、レベル ”

(*1) 波形表示データと波形表示フルデータについて
波形表示データ： 表示範囲のデータに対して 1024 ポイントの間引き処理を行ったデータ
波形表示フルデータ：表示範囲のデータに対して間引き処理を行っていないデータ

5.6.6 SCALE コマンド

表 5-13　SCALE コマンド

機能	コマンド	パラメータ	クエリ
波形表示選択	:DISPlay[:WINDow]:TRACe	SPECtrum|COHerence	SPEC|COH
オート・スケール	:DISPlay[:WINDow]:TRACe:ALL[:SCALe]:AUTO	-	×
表示上限値	:DISPlay[:WINDow]:TRACe:Y[:SCALe]:RLEVel	<real>	<real>
レベル・スケール	:DISPlay[:WINDow]:TRACe:Y[:SCALe]:PDIVision	10|5|2|1|0.5	10|5|2|1|0.5
表示パワー・ユニット	:UNIT:POWer	LINear|LOGarithmic	LIN|LOG
グリッド表示	:DISPlay[:WINDow]:TRACe:GRAPhics:GRID[:STATe]	ON|OFF	ON|OFF

5.6.7 APPLICATION コマンド

表 5-14　APPLICATION コマンド

機能	コマンド	パラメータ	クエリ
スーパー・インポーズ	:DISPlay[:WINDow]:TRACe:SUPErimpose	ON|OFF	ON|OFF
ピーク・ノーマライズ ON/OFF	:CALCulate2:LEVel:TYPE	POWer|NORMalize	POW|NORM
ドミナント ON/OFF	:CALCulate2:DOMInant[:STATe]	ON|OFF	ON|OFF
2 画面表示 ON/OFF	:DISPlay[:WINDow]:DUAL[:STATe]	ON|OFF	ON|OFF
2 画面表示 上画面選択	:DISPlay[:WINDow]:TRACe	SPECtrum|COHerence	SPEC|COH
リスト表示選択	:DISPlay[:WINDow]:LIST[:STATe]	ON|OFF	ON|OFF
リスト・カーソル位置	:DISPlay[:WINDow]:LIST:CURRent	<int>	<int>
全画面リスト表示	:DISPlay[:WINDow]:LIST:ALL	ON|OFF	ON|OFF
表示リスト選択	:DISPlay[:WINDow]:LIST:TYPE	SPECtrum|COHerence	SPEC|COH
コヒーレンシー			
2nd ピーク検索最小位置	{:MEASure|:FETCh}[:SCALar]:LENGth:COHerency:PEAK:RANGe:STARt	<real>	<real>
2nd ピーク検索最大位置	{:MEASure|:FETCh}[:SCALar]:LENGth:COHerency:PEAK:RANGe:STOP	<real>	<real>
2nd ピーク検索範囲	{:MEASure|:FETCh}[:SCALar]:LENGth:COHerency:PEAK:RANGe	“<real>,<real>= スタート値、ストップ値 ”	“<real>,<real>= スタート値、ストップ値 ”
α 位置指定	{:MEASure|:FETCh}[:SCALar]:LENGth:COHerency:POINt	<int>	<int>
α 値データ	{:MEASure | FETCh } [:SCALar]:LENGth:COHerency:ALPHa?	×	“<real>,<real>= 距離、レベル ”
β 値データ	{:MEASure | FETCh } [:SCALar]:LENGth:COHerency:BETA?	×	“<real>,<real>= 距離、レベル ”
半値幅表示			
半値幅表示選択	:CALCulate3:BANDwidth[:STATe]	ON|OFF	ON|OFF
半値幅演算選択	:CALCulate3:BANDwidth:TYPE	0|1|2|3=Pk-XdB|Envelope|RMS|Peak RMS	0|1|2|3=Pk-XdB|Envelope|RMS|Peak RMS
半値幅演算 XdB パラメータ	:CALCulate3:BANDwidth:XDB	<real>	<real>
半値幅演算 YdB パラメータ	:CALCulate3:BANDwidth:YDB	<real>	<real>
半値幅演算 K パラメータ	:CALCulate3:BANDwidth:K	<real>	<real>
半値幅演算 Kr パラメータ	:CALCulate3:BANDwidth:KR	<real>	<real>
半値幅データの出力要求	:CALCulate3:BANDwidth:DATA?	×	“<real>,<real>,<int>= 中心波長、バンド幅、バンド幅内のピーク本数 ”
リスト・データ			
データ数	:CALCulate3:POINts?	×	<int>
データ出力	:CALCulate3:DATA?	×	*FORMAT
トータル・パワー出力	:CALCulate2:TPOWer[:DATA]?	×	<real>

5.6.8 CURSOR コマンド

表 5-15 CURSOR コマンド

機能	コマンド	パラメータ	クエリ
X1 カーソル			
ON/OFF	:DISPlay[:WINDow]:MARKer1:STATe	ON|OFF	ON|OFF
波長	:DISPlay[:WINDow]:MARKer1:WAVelength	<real>	<real>
X2 カーソル			
ON/OFF	:DISPlay[:WINDow]:MARKer2:STATe	ON|OFF	ON|OFF
波長	:DISPlay[:WINDow]:MARKer2:WAVelength	<real>	<real>
Y1 カーソル			
ON/OFF	:DISPlay[:WINDow]:MARKer3:STATe	ON|OFF	ON|OFF
レベル	:DISPlay[:WINDow]:MARKer3:POWer	<real>	<real>
Y2 カーソル			
ON/OFF	:DISPlay[:WINDow]:MARKer4:STATe	ON|OFF	ON|OFF
レベル	:DISPlay[:WINDow]:MARKer4:POWer	<real>	<real>
カーソル表示選択	:DISPlay[:WINDow]:MARKer:ACTIve	UPper|LOWer	UP|LOW
全カーソル・オフ	:DISPlay[:WINDow]:MARKer:AOFF	-	×
カーソル・モード	:DISPlay[:WINDow]:MARKer:MODE	NORMal|DELTa	NORM|DELT = ノーマル・カーソル 1Δ カーソル
カーソル・データ出力	:DISPlay[:WINDow]:MARKer:DATA?	-	<real>,<real>,<real>, <real>,<real>,<real> (*1)
カーソル 2nd ピーク	:DISPlay[:WINDow]:MARKer:MAXimum:SEcond	-	×
カーソル・レフト・ピーク	:DISPlay[:WINDow]:MARKer:MAXimum:LEFT	-	×
カーソル・ライト・ピーク	:DISPlay[:WINDow]:MARKer:MAXimum:RIGHt	-	×

(*1) カーソル・データ・モードにより出力データが変わります。
ノーマル・モード： X1 カーソルの波長、X1 カーソルのレベル、X2 カーソルの波長、X2 カーソルのレベル、Y1 カーソルのレベル、Y2 カーソルのレベル
デルタ・モード： X1 カーソルの波長、X1 カーソルのレベル、X1 と X2 カーソルの波長差、X1 と X2 カーソルのレベル差、Y1 カーソルのレベル、Y2 カーソルのレベル

5.6.9 SYSTEM コマンド

表 5-16　SYSTEM コマンド

機能	コマンド	パラメータ	クエリ
プリセット	:SYSTem:PRESet		
ラベル入力	:DISPlay[:WINDow]:TEXT:DATA	<str>	<str>
時刻設定	:SYSTem:DATE	“<int>,<int>,<int>= 年、月、日 ”	“<int>,<int>,<int>= 年、月、日 ”
日付設定	:SYSTem:TIME	“<int>,<int>= 時、分 ”	“<int>,<int>= 時、分 ”
システム・シャットダウン	:SYSTem:HALT	-	×
オプションの有無	:SYSTem:OPTion?	×	0|1= オプション無 | オプション有

5.6.10 SAVE/LOAD コマンド

表 5-17　SAVE/LOAD コマンド

機能	コマンド	パラメータ	クエリ
セーブ	:FILE:STORe	<int> or <str>	×
ロード	:FILE:LOAD	<int> or <str>	×

5.6.11 GPIB 機能コマンド

表 5-18 GPIB 機能コマンド

機能	コマンド	パラメータ（= 概要）	クエリ（= 概要）
出力データ・フォーマット フォーマット選択	:FORMat:DATA	ASCii|REAL,{32|64} =ASCii : 数値データ =REAL : バイナリ・データ *1	ASC|REAL,32|REAL,64
バイナリ・フォーマット	:FORMat:BORDer	SWAPped|NORMal	SWAP|NORM
ASCII・Y 軸 出力データ桁数選択	:FORMat:ASCii[:Y]:DATA	0|1=4 桁 |6 桁	<int>
オペレーション・ステータス・イネーブル・レジスタの設定、読み出し	:STATus:OPERation:ENABle	<int>=0 ～ 65535	<int>
クエスチョナブル・ステータス・イネーブル・レジスタの設定、読み出し	:STATus:QUEStionable:ENABle	<int>=0 ～ 65535	<int>
デバイス・ステータス・イネーブル・レジスタの設定、読み出し	:STATus:DEVice:ENABle	<int>=0 ～ 65535	<int>
スタンダード・オペレーション・ステータス・レジスタの読み出し	:STATus:OPERation[:EVENt]?	-	<int>=0 ～ 65535
クエスチョナブル・ステータス・レジスタの読み出し	:STATus:QUEStionable[:EVENt]?	-	<int>=0 ～ 65535
デバイス・ステータス・レジスタの読み出し	:STATus:DEVice[:EVENt]?	-	<int>=0 ～ 65536
ステータス・バイト・クリア	:STATus:PRESet	-	-
エラー読み出し	:SYSTem:ERRor?	-	<int>,<str>
エラー・キューすべての読み出し	:SYSTem:ERRor:ALL?	-	<int>,<str>, ...

(*1) データ・フォーマットが 32 ビットか 64 ビットかを選択します。

5.6.12 GPIB/LAN 説明例題

Visual Basic6.0 を用いた GPIB/LAN 転送プログラムの例題を説明します。

例題の内容は以下のとおりです。

- 掃引開始
- spectrum のピーク波長と、そのレベルおよび半値幅の読み出し
- coh 解析時の α/β 値の読み出し
- spctrum 波形／ coh 波形の x 軸／ y 軸の読み出し

1. 初期化
   ↓
2. ブロック・データ転送モード指定
   ↓
3. 測定設定条件および表示条件設定
   ↓
4. 掃引開始
   ↓
5. 掃引終了
   ↓
6. スペクトラム・ピーク波長と、そのレベルの読み出し
   ↓
7. スペクトラム波形の半値幅の読み出し
   ↓
8. Coherence データの α/β 値の読み出し
   ↓
9. spectrum 波形の x 軸／ y 軸の読み出し
   ↓
10. coherence 波形の x 軸／ y 軸の読み出し

5.6.12.1 GPIB 例題

```
'GPIB 例題
Dim A8341 As Integer, OSAud As Integer
Dim GpConfig As Integer, boardID As Integer
Dim BinType As Integer


'MAIN
Private Sub Meas8341_GPIB()

Dim strBuf As String
Dim P As Integer
Dim BitLength As String
Dim SetCenter As Double, SetSpan As Double
Dim PeakWavelength As Double, PeakPower As Double
Dim SpeCent As Double, SpeBW As Double
Dim CohAlphaMag As Double, CohAlphaLen As Double, CohBetaMag As Double, CohBetaLen As Double
Dim SpeRedX() As Double, CohRedX() As Double
Dim WfmSpeY() As Double, WfmCohY() As Double
Dim NspectRed As Integer, NcohRed As Integer


  'GPIB IF 初期化
  A8341 = 8
  GpConfig = False
  Call Init_GPIB


  'GPIB Device configuration:No Byte Swap
  Call ibconfig(OSAud, IbcReadAdjust, 0)


  'RESET
  Call ibwrt(OSAud, "*RST")


  ' スペクトル画面に設定
  Call ibwrt(OSAud, ":DISP:TRAC SPEC")


  ' 波形データの送出をバイナリ転送に指定、バイトオーダーをスワップに設定
  BinType = 1       '0=64bit, i=32bit
  BitLength = "32"
  Call ibwrt(OSAud, ":FORM:DATA REAL," + BitLength + ";BORD SWAP")


  'REF LEVEL の設定 -10dBm
  Call ibwrt(OSAud, ":DISP:TRAC:Y:RLEV -10")


  ' コヒーレンス・ピークサーチ範囲の設定 3mm から 7mm
  Call ibwrt(OSAud, ":MEAS:LENG:COH:PEAK:RANG 0.003,0.007")


  ' センタ・スパンを設定
  SetCenter = "660"
  SetSpan = "50"
  Call ibwrt(OSAud, ":DISP:TRAC:X:CENT " & Format(SetCenter) & "E-9")
  Call ibwrt(OSAud, ":DISP:TRAC:X:SPAN " & Format(SetSpan) & "E-9")


  ' 半値幅演算を XdB に設定
  Call ibwrt(OSAud, ":CALC3:BAND ON")
  Call ibwrt(OSAud, ":CALC3:BAND:TYPE 0")


  ' ステータス・クリア & シングル測定コマンド
  Call ibwrt(OSAud, "*CLS")
  Call ibwrt(OSAud, "INIT:CONT OFF;IMM")
```

```
' 測定終了を確認・測定が終了すれば "1" が戻る
strBuf = Space(5)
Call ibwrt(OSAud, "*OPC?")
Call ibrd(OSAud, strBuf)
P = InStr(strBuf, Chr(10))

' ピーク・サーチ・データの取得
' ピーク波長出力要求
Call ibwrt(OSAud, ":MEAS:PEAK?")
strBuf = Space(50)
Call ibrd(OSAud, strBuf)
' データの分離と丸め
P = InStr(strBuf, ",")
PeakWavelength = Int(Val(Left(strBuf, P - 1)) * 1000000000000# + 0.5) / 1000
PeakPower = Int(Val(Mid(strBuf, P + 1)) * 1000 + 0.5) / 1000

' 半値幅演算のスペクトル幅を取得
' スペクトル幅データ出力要求
Call ibwrt(OSAud, ":CALC3:BAND:DATA?")
strBuf = Space(50)
Call ibrd(OSAud, strBuf)
' データの分離と丸め
P = InStr(strBuf, ",")
SpeCent = Int(Val(Left(strBuf, P - 1)) * 1000000000000# + 0.5) / 1000
SpeBW = Int(Val(Mid(strBuf, P + 1)) * 1000000000000# + 0.5) / 1000

' アルファ取得
'  α出力要求
Call ibwrt(OSAud, ":MEAS:LENG:COH:ALPH?")
strBuf = Space(50)
Call ibrd(OSAud, strBuf)
'  データの分離と丸め
P = InStr(strBuf, ",")
CohAlphaLen = Int(Val(Left(strBuf, P - 1)) * 1000000# + 0.5) / 1000
CohAlphaMag = Int(Val(Mid(strBuf, P + 1)) * 1000# + 0.5) / 1000

'  ベータ取得
'  β出力要求
Call ibwrt(OSAud, ":MEAS:LENG:COH:BETA?")
strBuf = Space(50)
Call ibrd(OSAud, strBuf)
'  データの分離と丸め
P = InStr(strBuf, ",")
CohBetaLen = Int(Val(Left(strBuf, P - 1)) * 1000000# + 0.5) / 1000
CohBetaMag = Int(Val(Mid(strBuf, P + 1)) * 1000# + 0.5) / 1000

' 波長軸データ取得
Call ibwrt(OSAud, ":DISP:TRAC:X:DATA? SPEC")
'  データのバイナリ受信
Call Read41_Binary(NspectRed, SpeRedX())

'  Y軸データ（レベル）を要求
Call ibwrt(OSAud, ":DISP:TRAC:Y:DATA? SPEC")
'  データのバイナリ受信
Call Read41_Binary(NspectRed, WfmSpeY())

' コヒーレンス波形データを読み取る
Call ibwrt(OSAud, ":DISP:TRAC:X:DATA? COH")
'  データのバイナリ受信
Call Read41_Binary(NcohRed, CohRedX())

'  Y軸データ（ビジビリティ）を要求
```

```
  Call ibwrt(OSAud, ":DISP:TRAC:Y:DATA? COH")
  ' データのバイナリ受信
  Call Read41_Binary(NcohRed, WfmCohY())


  ' 測定終わり
  End_GPIB

End Sub




'GPIB IF 初期化
Public Sub Init_GPIB()

  boardID = 0

  If GpConfig = False Then
  Call ibdev(boardID, A8341, 0, T3s, 1, 0, OSAud)       'time out
  Call ibtmo(OSAud, T3s)
  End If

End Sub




'GPIB IF 終了
Public Sub End_GPIB()

Call ibsic(OSAud)
Call ibonl(OSAud, 0)

End Sub




' データのバイナリ受信
Private Sub Read41_Binary(n As Integer, Data() As Double)
  '8341 から、Binary format で送出された波形データを受信する

  Dim i As Integer, intBuf As Integer, P As Integer, NByte As Integer
  Dim strS As String, strBuf As String
  Dim dblBUFFER() As Double, sglBUFFER() As Single

  ' バイナリ・データのヘッダ部を読み取る
  Call ibrd32(OSAud, intBuf, 1)
  If Chr(intBuf) = "#" Then
    Call ibrd32(OSAud, intBuf, 1)
    strS = ""
    For i = 1 To Val(Chr(intBuf))
      Call ibrd32(OSAud, intBuf, 1)
      strS = strS + Chr(intBuf)
    Next i
  End If

  ' 転送バイト数
  NByte = Val(strS)

  ' バイナリ・ブロック・データを取得
  Select Case BinType
    Case 0    ' 64bit を指定した場合
      n = NByte / 8
      Erase dblBUFFER()
      ReDim dblBUFFER(0 To n - 1), Data(0 To n - 1)
      Call ibrd32(OSAud, dblBUFFER(0), NByte)

    Case 1    ' 32bit を指定した場合
      n = NByte / 4
      Erase sglBUFFER()
```

```
        ReDim sglBUFFER(0 To n - 1), Data(0 To n - 1)
        Call ibrd32(OSAud, sglBUFFER(0), NByte)
    End Select


    ' 最後のデリミタ (LF) を受信する
    strBuf = Space(2)
    Call ibrd(OSAud, strBuf)
    P = InStr(strBuf, Chr(10))

    ' バッファ・データをコピーして戻す
    Select Case BinType
      Case 0
        For i = 0 To n - 1
         Data(i) = dblBUFFER(i)
        Next i
      Case 1
        For i = 0 To n - 1
         Data(i) = sglBUFFER(i)
        Next i
    End Select

End Sub
```

5.6.12.2 LAN 例題

```
'LAN 例題
Dim BinType As Integer


'MAIN
Private Sub Meas8341_TCP()

Dim strBuf As String
Dim P As Integer
Dim BitLength As String
Dim SetCenter As Double, SetSpan As Double
Dim GrhDtByteLen As Integer
Dim PeakWavelength As Double, PeakPower As Double
Dim SpeCent As Double, SpeBW As Double
Dim CohAlphaMag As Double, CohAlphaLen As Double, CohBetaMag As Double, CohBetaLen As Double
Dim SpeRedX() As Double, CohRedX() As Double
Dim WfmSpeY() As Double, WfmCohY() As Double
Dim NspectRed As Integer, NcohRed As Integer


  'RESET
  tcpClient.SendData "*RST" & vbCrLf


  ' 初期化終了を確認。終了すれば "1" が戻る
  tcpClient.SendData "*OPC?" & vbCrLf
  strBuf = Space(10)
  Call ReadData_TCP(strBuf)
  P = InStr(strBuf, Chr(10))


  ' スペクトル画面に設定
  tcpClient.SendData ":DISP:TRAC SPEC" & vbCrLf


  ' 波形データの送出をバイナリ転送に指定、バイト・オーダをスワップに設定
  BinType = 1           '0=64bit, i=32bit
  BitLength = "32"
  tcpClient.SendData ":FORM:DATA REAL," + BitLength + ";BORD SWAP" & vbCrLf


  'REF LEVEL を "-10dBm" に設定
  tcpClient.SendData ":DISP:TRAC:Y:RLEV -10" & vbCrLf


  ' コヒーレンス・ピーク・サーチ範囲の設定 3mm から 7mm
  tcpClient.SendData ":MEAS:LENG:COH:PEAK:RANG 0.003,0.007" & vbCrLf


  ' センタ・スパンを設定
  SetCenter = "660"
  SetSpan = "50"
  tcpClient.SendData ":DISP:TRAC:X:CENT " & Format(SetCenter) & "E-9" & vbCrLf
  tcpClient.SendData ":DISP:TRAC:X:SPAN " & Format(SetSpan) & "E-9" & vbCrLf


  ' 半値幅演算を XdB に設定
  tcpClient.SendData ":CALC3:BAND ON" & vbCrLf
  tcpClient.SendData ":CALC3:BAND:TYPE 0" & vbCrLf


  ' ステータス・クリア & シングル測定コマンド
  tcpClient.SendData "*CLS" & vbCrLf
  tcpClient.SendData "INIT:CONT OFF;IMM" & vbCrLf


  ' 測定終了を確認・測定が終了すれば "1" が戻る
  tcpClient.SendData "*OPC?" & vbCrLf
  strBuf = Space(10)
  Call ReadData_TCP(strBuf)
  P = InStr(strBuf, Chr(10))
```

```
' ピーク・サーチ・データの取得
' ピーク波長出力要求
tcpClient.SendData ":MEAS:PEAK?" & vbCrLf
strBuf = Space(50)
Call ReadData_TCP(strBuf)
' データの分離と丸め
P = InStr(strBuf, ",")
PeakWavelength = Int(Val(Left(strBuf, P - 1)) * 1000000000000# + 0.5) / 1000
PeakPower = Int(Val(Mid(strBuf, P + 1)) * 1000 + 0.5) / 1000


' 半値幅演算のスペクトル幅を取得
' スペクトル幅データ出力要求
tcpClient.SendData ":CALC3:BAND:DATA?" & vbCrLf
strBuf = Space(50)
Call ReadData_TCP(strBuf)
' データの分離と丸め
P = InStr(strBuf, ",")
SpeCent = Int(Val(Left(strBuf, P - 1)) * 1000000000000# + 0.5) / 1000
SpeBW = Int(Val(Mid(strBuf, P + 1)) * 1000000000000# + 0.5) / 1000


' アルファ取得
'α出力要求
tcpClient.SendData ":MEAS:LENG:COH:ALPH?" & vbCrLf
strBuf = Space(50)
Call ReadData_TCP(strBuf)
' データの分離と丸め
P = InStr(strBuf, ",")
CohAlphaLen = Int(Val(Left(strBuf, P - 1)) * 1000000# + 0.5) / 1000
CohAlphaMag = Int(Val(Mid(strBuf, P + 1)) * 1000# + 0.5) / 1000


' ベータ取得
'β 出力要求
tcpClient.SendData ":MEAS:LENG:COH:BETA?" & vbCrLf
strBuf = Space(50)
Call ReadData_TCP(strBuf)
' データの分離と丸め
P = InStr(strBuf, ",")
CohBetaLen = Int(Val(Left(strBuf, P - 1)) * 1000000# + 0.5) / 1000
CohBetaMag = Int(Val(Mid(strBuf, P + 1)) * 1000# + 0.5) / 1000


' 波長軸データ取得
GrhDtByteLen = GetBinDataLen()
tcpClient.SendData ":DISP:TRAC:X:DATA? SPEC" & vbCrLf
' データのバイナリ受信
Call BinaryDataRead_IP(NspectRed, SpeRedX(), GrhDtByteLen)


'Y 軸データ（レベル）を要求
GrhDtByteLen = GetBinDataLen()
tcpClient.SendData ":DISP:TRAC:Y:DATA? SPEC" & vbCrLf
' データのバイナリ受信
Call BinaryDataRead_IP(NspectRed, WfmSpeY(), GrhDtByteLen)


' コヒーレンス波形データを読み取る
GrhDtByteLen = GetBinDataLen()
tcpClient.SendData ":DISP:TRAC:X:DATA? COH" & vbCrLf
' データのバイナリ受信
Call BinaryDataRead_IP(NcohRed, CohRedX(), GrhDtByteLen)


'Y 軸データ（ビジビリティ）を要求
GrhDtByteLen = GetBinDataLen()
tcpClient.SendData ":DISP:TRAC:Y:DATA? COH" & vbCrLf
' データのバイナリ受信
Call BinaryDataRead_IP(NcohRed, WfmCohY(), GrhDtByteLen)
```

```
  ' 測定終わり
  Call CloseTCP

End Sub



' ソケットの初期化とコネクション処理
Private Sub ConnectTCP()

  tcpClient.RemoteHost = "x.x.x.x"                ' 8341 の IP Address をセットします
  tcpClient.Protocol = sckTCPProtocol             ' protocol に TCP を指定します
  tcpClient.RemotePort = 5025                     ' 8341 の Remote port No として "5025" をセットします
  tcpClient.LocalPort = 0                         ' コントローラ側の  Local port No を "0" にセットします
  tcpClient.Connect                               ' 8341 の Remote port へコネクトします

End Sub


' コネクションの終了処理
Private Sub CloseTCP()

  tcpClient.Close

End Sub


' クエリ応答データ読み込み
Private Sub ReadData_TCP(readbuff As String)

  Do While (tcpClient.BytesReceived = 0)
    DoEvents
  Loop

  tcpClient.GetData readbuff

End Sub



' バイナリ・データの取得
Private Sub BinaryDataRead_IP(datanum As Integer, databuff() As Double, datalimit As Integer)
  Dim cnt As Integer, kk As Integer
  Dim ii As Integer, jj As Integer
  Dim bytedata As Byte
  Dim bytbuf() As Byte

  DoEvents
  Do While (tcpClient.BytesReceived < datalimit)
    DoEvents

  Loop

  cnt = tcpClient.BytesReceived
  tcpClient.GetData bytedata, vbByte              ' "#"
  tcpClient.GetData bytedata, vbByte              ' "1" - "9"
  ii = CInt(Chr(bytedata))

  For jj = 0 To ii - 1
    tcpClient.GetData bytedata, vbByte            ' read data of binary data length
  Next jj

  kk = 0
  Select Case BinType
  Case 0    '64bit format
    ReDim databuff((cnt - 2 - ii) / 8)
    For jj = 1 To (cnt - 2 - ii) / 8
      tcpClient.GetData databuff(jj), vbDouble
      kk = kk + 1
    Next
```

```
  Case 1    '32bit format
    ReDim databuff((cnt - 2 - ii) / 4)
    For jj = 1 To (cnt - 2 - ii) / 4
      tcpClient.GetData sngBuf, vbSingle
      databuff(jj) = CDbl(sngBuf)
      kk = kk + 1
    Next
  End Select

  If tcpClient.BytesReceived > 0 Then
    strBuf = Space(tcpClient.BytesReceived + 1)
    tcpClient.GetData strBuf
  End If

  datanum = kk

End Sub



'グラフ・データ長を取得
Private Function GetBinDataLen() As Integer
Dim strBuf As String
Dim DataLen As Integer

  GetBinDataLen = 0

  tcpClient.SendData ":DISP:TRAC:POIN?" & vbCrLf   'データ数要求
  strBuf = Space(20)
  Call ReadData_TCP(strBuf)                        'データ数受信
  DataLen = CInt(strBuf)

  Select Case BinType
  Case 0  '64bit format
    GetBinDataLen = DataLen * 8
  Case 1  '32bit format
    GetBinDataLen = DataLen * 4
  End Select

End Function



'初期設定
Private Sub Form_Load()
  Call ConnectTCP
End Sub
```